大規模線形計画問題における RPPアルゴリズムの有効性について

植松康祐* 百武稔郎** 成久洋之

*岡山理科大学 情報処理センター

**岡山理科大学 電子理学科

(昭和62年9月30日 受理)

1. はじめに

線形計画法 (Linear Programming, 略してLP) とは, 線形の不等式で与えられた制約条件の下で, 線形の目的関数を最大又は最小にするものである. その適用範囲は幅広く, 工業, 商業, 公共政策, 軍事などにおよび, 経営科学あるいはオペレーションズ・リサーチ (Operations Research) の実用分野において著しい成果をあげている.

線形計画問題の解法は, 最初1948年から1952年にかけて開発された. この時のDantzig〔1〕による単体法 (Simplex Method) が, 現在まで, 中心的な地位を占めてきている. この時期に初期のコンピュータを利用する試みがなされている. しかし, これらの試みは規模においては限られたものであった. おそらく, 一般的な100次の行列を扱うことは大変なことであった. 今日では, 100個の制約条件を持つLP問題は小さいものと考えられる.

LPモデルには, 次のような重要な特徴がある. 比較的簡単なモデルについてさえ, 式と変数の個数が大きくなる. そして処理されるデータの量や計算量は, ストレージ空間, 計算時間, 信頼性および精度の点で, コンピュータをもってしても, 驚異的に多くなるということである. 現実に, 科学技術向上と伴って, 製品の高品質化や事業の拡大などにより, LPの適用対象の規模が巨大化している. 条件式や変数の個数が数千個といった大規模なLP問題も珍しくはない.

これらの大規模問題解決のための研究は, 1959年, DantzigとWolfe〔2〕を初めとして, 色々な分割法が提案されている. これらの方法は, 係数行列において0の占める割合が非常に多い場合のみ有効である. また, 使用するにはかなりの経験がいるという難点もある.

そこで我々は, 1984年Sethi and Thompson〔3〕のPivot and Probe Algorithm

（以下PPアルゴリズムと略す）に注目した．

PPアルゴリズムとは最適解が全条件式によって与えられるのではなく，実行可能領域を形式する一部の条件式によって決められることに着目して，最適解の求解に対し有効な条件式のみを抽出して補助問題を作成し，必要に応じて適当な条件式を付加してゆくことにより最終的に最適解を求めてゆこうとするものである．このPPアルゴリズムの有効性は従来の解法と比べかなり有効性が示され，また，改訂単体法の有効性が見られる範囲においてもその有効性が判明した．

我々はPPアルゴリズムを改良して，さらに大規模問題における有効性を発揮するRPPアルゴリズムを提案する．

2．RPPアルゴリズム

2．1　RPPアルゴリズム概要

一般に線形計画問題を解いてゆく場合，シンプレックス法により解かれる．これは，与えられた線形不等式条件式が満たす実行可能領域において，一つの実行可能基底解を見つけ，それを出発点とし，隣接する実行可能基底解に移行しながら有限回の移行により目的関数を最適化とする実行可能基底解に到達するものである．しかし，シンプレックス法では，線形計画問題の規模が大きくなるに従い掃き出し要素が増え，著しく処理時間が増加するなどの問題が出てくる．これらの問題を解決するために関発された改訂単体法は，条件式数に比べ変数の数が多い場合に有効であるが，それ以外の場合に対しては必ずしも有効でない．

そこで，線形計画問題における最適解は全条件式に関係しているものではなく，一般的に，その有効な条件式は問題の種類によって異なるが，5～30%程度の条件式が最適解に関与しているものとされている．したがって，線形計画問題の求解に対しては，その5～30%の有効な条件式について解けばよいわけである．

このことから，与えられた全条件式について解いてゆくのではなく，有効な条件式を線形計画問題の中から選び出し，初期補助問題を作成してシンプレックス法で解き，必要に応じてさらに有効な条件式に付加してゆくことにより，シンプレックス法と比較してかなりの処理時間の短縮が期待できる．

最適解は，全線形不等式条件が満たす凸多面体の実行可能領域の端点に存在する特徴を持つ．LP問題の中には，必ずしも一意の端点に存在しないなどの退化した問題があるが，特殊な場合であるのでここでは省く．そこで，最適解が端点に存在することから，実行可能領域を形成する条件式に着目する．その条件式は明らかに最適解の求解に対して有効であると言える．

次のようなLP問題が与えられているものとする．

目的関数　$z=\sum_{j=1}^{m} c_j x_j \longrightarrow max$

制約条件式　$\sum_{j=1}^{n} a_{ij} x_j \leqq b_i \quad (\pi)$

$x_j \geqq 0 \quad i \in I \quad j \in J$

条件式の添字集合：$I=\{1, 2, 3, \cdots, m\}$

変数の添字集合　：$J=\{1, 2, 3, \cdots, n\}$

初期補助問題において，最適解を含み実行可能領域を形成する条件式，すなわち各座標軸上で原点に近い接辺を持つ条件式を選び出す．

それを添字集合で表わすと

$$K=\underset{j\in J}{U}\{i \mid d_{ij}=\min_{h} b_h / a_{hj} \quad \text{where} \quad i, h \in I, j \in J, \quad a_{hj}>0\} U K_0$$

ここで，この問題(π)に，便宜上次の$(m+1)$番目の条件式を付加する．

$$\sum_{j=1}^{n} x_j \leqq M$$

ただし，Mは非常に大きな値であり，問題(π)に有限な解を与えることを保障するものである．

$$K_0=\{m+1\}$$
$$\bar{K}=I \setminus K \quad (I \cap \bar{K})$$

これらの添字集合をもとに初期補助問題(π^s)を定義する．

目的関数　：　$z=\sum_{j=1}^{n} c_j x_j \longrightarrow max$

制約条件式　：　$\sum_{j=1}^{n} a_{ij} x_j \leqq b_i \quad (\pi^s)$

$x_j \geqq 0 \quad i \in K, \quad j \in J$

この補助問題(π^s)は明らかに，原問題(π)より少ない条件式数から構成されている．また，もし原問題(π)が解をもつならば，補助問題(π^s)が便宜上の条件式を含んでいることから解を持つ．そこで，この補助問題(π^s)をシンプレックス法で解き，補助問題の解を得る．それが，原問題(π)の最適解でない場合，目的関数値を最適解に近づけてゆくために添字集合$\bar{K}$に属する条件式の中で，有効な条件式を複数個補助問題へ導入し，新たな補助問題として解いてゆく．

新たに補助問題へ付加する条件式の探索を以下に説明する．凸集合において，それに属する任意の2点を取り出し，その2点間を結ぶ線分上の点はかならずその凸集合に属する．また最適解は，凸多面体の実行可能領域の端点に存在することから，補助問題(π^s)の実行可能基底解と原問題(π)の実行可能解との2点間の線分上に添字集合$\bar{K}$に属する

条件式が存在するとこれに矛盾し，その時点での補助問題(π^s)の目的関数値は，原問題(π)の最適解に達していないことになる．さらに，最適解に近づけてゆくために，その2点間の線分上において，原問題(π)の実行可能解に最も近い交点を持つ条件式，すなわち実行可能領域を形成する有効な条件式を抽出し，補助問題(π^s)へ付加する．その式は以下の式によって導き出される．

原問題(π)の実行可能解x_Dと補助問題(π^s)の実行可能基底解x_p間の線分上の点をxとすると

$$x=(1-\lambda)x_p+\lambda x_D \qquad \because \lambda\in[0,\ 1]$$

$$a_h x=(1-\lambda_h)a_h x_h+\lambda a_h x_D \qquad \because h\in\bar{K}$$

これを，λ_hについて解くと

$$\lambda_h=(b_h-a_h x_p)/(a_h x_D-a_h x_p)$$

上式より，$a_h x_D>b_h$のとき，すなわち，h条件式を満足した場合λ_hは［0，1）間に存在する．原問題(π)の実行可能領域を形成する条件式を選択するには，原問題の実行可能解x_pに最も近い交点を求めればよいので，最小なλ_hを決定する．

$$\lambda_i=min\ \lambda_h\ (i,\ h\in H)$$

$$H=\{h\in\bar{K}\mid a_h x_D>b_h\}$$

これより求められた有効条件式を新たに補助問題へ導入する．

条件式iの交点$\tilde{x}$は

$$\tilde{x}=(1-\lambda_i)x_p+\lambda_i x_D$$

で求められる．

ここで，複数の条件式が最小値λ_iを持つならば，その条件式全部を補助問題に導入する．

また，目的関数の最大係数を指標として，さらに最適解の求解に対して有効な条件式を抽出する．目的開数の係数が大きいほど，言いかえれば目的関数の単位当りのコスト係数が大きいほど変数が1単位増えたとき目的関数値の増加分が大きく，最適解に近づいてゆくには有効である．この係数に注目して大きい値から順に任意の個数を選び出し，それに対応する条件式を補助問題に導入する．このような操作を繰り返し，λ_iが［0，1）をとらないとき有効条件式が存在しないことになり，その時点における補助問題の解が原問題(π)の最適解となる．

2．2 RPPアルゴリズム

次のようなLP問題が与えられているものとする．

目的関数 ： $z=\sum_{j=1}^{n} c_j x_j \longrightarrow max$

制約条件式　：　$\sum_{j=1}^{n} a_{ij} x_j \leqq b_i$

$$x_j \geqq 0 \qquad (\pi)$$

$$i \in I,\quad j \in J$$

この問題(π)に便宜上次の$(m+1)$番目の条件式を付加する．

$$\sum_{j=1}^{n} x_j \leqq M$$

ただしMは非常に大きな値とする．

〔Notation〕

条件式の添字集合：$I=\{1,\ 2,\ 3,\ \cdots,\ m\}$

変数の添字集合　：$J=\{1,\ 2,\ 3,\ \cdots,\ n\}$

補助問題条件式の添字集合

$$K_0=\{m+1\}$$

$$K_1=\underset{j\in J}{U}\{i \mid d_{ij}=\min_{h} b_h/a_{hj} \quad \text{where} \quad i,\ h\in I,\ j\in J,\ a_{hj}>0\}\ U\ K_0$$

$$\bar{K}_i=I \setminus K_i \quad (I \supset K_i)$$

$$a_h=(a_{h1},\ a_{h2},\ \cdots,\ a_{hn})$$

〔Algorithm〕

（1）　初期値を $N=1$とする．

添字集合 K_Nより補助問題を作成する．

目的関数　：　$z=\sum_{j=1}^{n} c_j x_j \longrightarrow max$

制約条件式　：　$\sum_{j=1}^{n} a_{ij} x_j \leqq b_i$

$$x_j \geqq 0 \qquad (\pi^s)$$

$$j \in J,\quad i \in K_N$$

この問題をシンプレックス法で解き，その解を$X(N)$とする．

（2）　原点と$X(N)$を結ぶ直線と$\sum_{j=1}^{n} a_{ij} x_j = b_i\ i \in \bar{K}_n$が交点を持つか調べる．もし持たないならば$X(N)$が最適解となり終了する．交点を持つならば次に進む．

（3）　もし交点Wが存在するとき

$a_h W=\lambda_h a_h x(N)$を満たすことにより

$\lambda\ h=b_h/a_h x(N)$となるλ_hが存在する．

$$E_N=\{\, i \mid \lambda_i=\min_h \lambda_h,\ \lambda_i\in[0,\ 1],\ h\in\bar{K}_N\}$$

（4） ある整数Pを選ぶ(処理時間を最小とする最適なPは問題のタイプによって変わる)

$$L_q=\{\, j \mid c_j=max\ c_k,\ k\in J\setminus\bigcup_{k=1}^{q=1} L_k\}\ q=1,\ 2,\ 3,\ \cdots,\ P$$

目的関数の係数の大きい順にP個の実行可能領域を形成する条件式を選び出す.

$$Y_N(i)=(1-\lambda_i)\,Y_{N-1}(i)+\lambda_i X(N)$$

ただし,

$$Y_0(j)=\{0,\cdots,\ 0, d_{ij},\ 0,\cdots,\ 0\}$$

$$d_{ij}=\min_h\ b_h/a_{hj},\quad h\in K_1,\quad j\in\bigcup_{q=1}^{P} L_q$$

$$F_N=\bigcup_s\{i \mid \lambda_i=\min_h(b_h-a_h Y_{N-1}(s))\}/(a_h X(N)-a_h Y_{N-1}(s)),$$

$$\lambda_i\in[0,1], h\in K_N\},\ s\in\bigcup_{q=1}^{P} L_q$$

$$K_{N+1}=K_N\cup(E_N\cup F_N)$$

N = N +1として(1)へ戻る.

2．3 具体例

ここで次の7式，2変数の線形計画問題を解く，5つの表はアルゴリズムに従って選び出された補助問題のシンプレックスタブローである．図1はそれにともなった条件式と実行解を示したものである.

目的関数　$z=-x_1-x_2\rightarrow max$

制約条件式
$$-x_1+3x_2\leqq 36$$
$$6x_1-2x_2\leqq 39$$
$$-x_1+10x_2\leqq 130$$
$$4x_1+x_2\leqq 46$$
$$3x_1+x_2\leqq 52$$
$$2x_1+3x_2\leqq 53$$
$$2x_1-x_2\leqq 12$$
$$x_1, x_2\geqq 0$$

表1は与えられたLP問題をコンデンスタブローに示したものである．次に処理時間を最小とする最適なPを選び（ここではP＝2）有効条件式を抽出し，その補助問題をタブロー化したものが表2である.このタブローをシンプレックス法で解き表3を得る.この補助問題での最適解は図1の点Aに示めされている．補助問題の最適解Aと原点0の線分上の交点のうち原点に最っとも近い交点Bをもつ条件式⑤とPにより選ばれる点C, Dとの線分上の点E, Fを持つ有効条件式③, ④を新たに補助問題に導入し表4とする．表4をデュアルシンプレックス法を用いて解き，表5，図1の点Gを得る．以

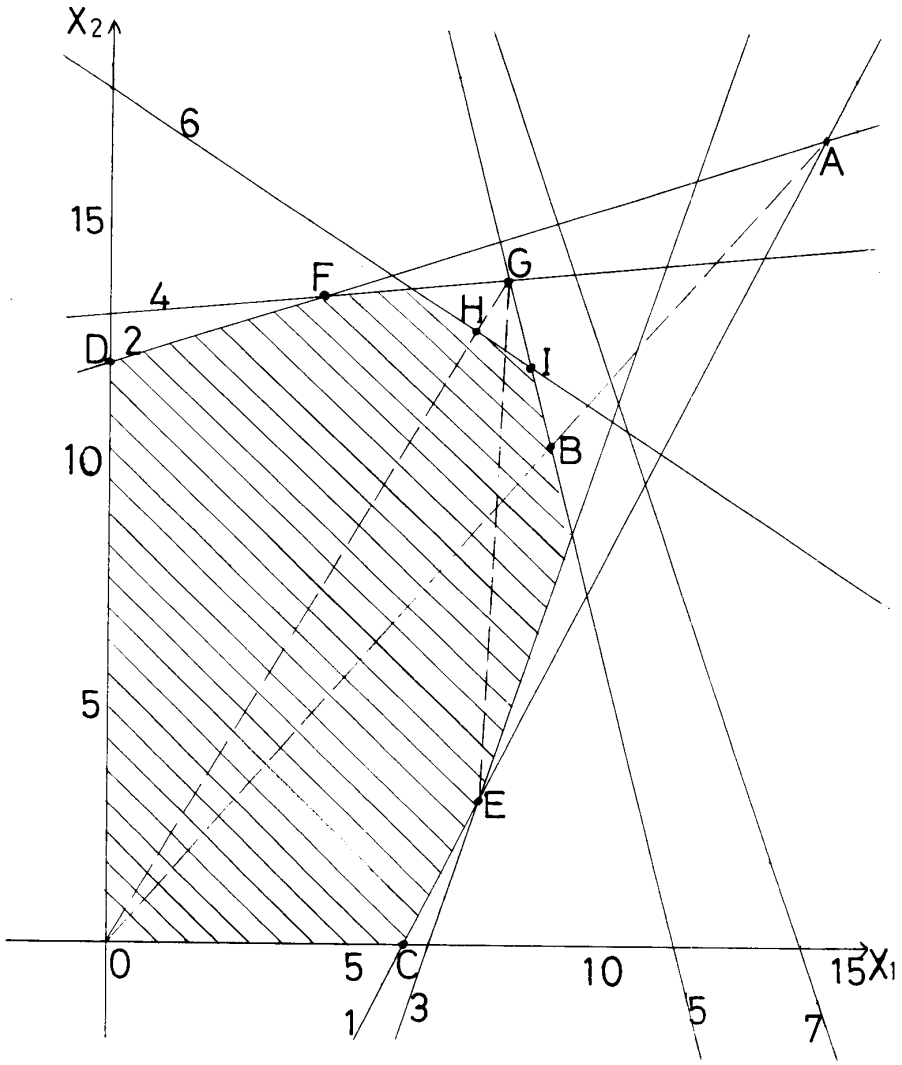

図1　具体例における条件式と実行解

Row	x_1	x_2	b	
1	−1	3	36	$=s_1$
2	6	−2	39	$=s_2$
3	−1	10	130	$=s_3$
4	4	1	46	$=s_4$
5	3	1	52	$=s_5$
6	2	3	53	$=s_6$
7	2	−1	12	$=s_7$
8	1	1	1.0E+10	$=s_8$
z	−1	−1	0	

表1　原問題におけるタブロー

	x_1	x_2	b
s_1	2	−1	12
s_2	−1	3	36
s_8	1	1	1.0E+10
z	−1	−1	0

表2　初期補助問題におけるタブロー

	s_1	s_2	b
x_1	0.6	0.2	14.4
x_2	0.2	0.4	16.8
s_8	−0.8	−0.6	1.0E+10
z	0.8	0.6	31.2

表3　初期補助問題における最適タブロー

	s_1	s_2	b
x_1	0.6	0.2	14.4
x_2	0.2	0.4	16.8
s_8	−0.8	−0.6	1.0E+10
s_5	−2.6	−1.2	−28.4
s_3	−3.2	−0.4	−13.8
s_4	−1.4	−3.8	−23.6
z	0.8	0.6	31.2

表4　有効条件式③④⑤を導入したタブロー

	s_5	s_4	b
x_1	0.24	−0.02	8.05
x_2	0.02	0.10	13.80
s_8	−0.28	−0.07	1.0E+10
s_1	−0.46	0.15	9.70
s_3	−1.42	0.34	18.31
s_2	0.17	−0.32	2.63
z	0.27	0.07	21.85

表5　更新された補助問題における最適タブロー

	s_5	s_6	b
x_1	0.31	−0.18	8.54
x_2	−0.22	0.47	12.05
s_8	−0.14	−0.36	1.0E+10
s_1	−0.84	0.64	7.08
s_3	−2.22	1.45	12.03
s_2	0.90	−1.36	8.57
s_4	2.30	−4.14	18.50
z	0.10	0.31	20.50

表6　原問題に対する最適タブロー

後この操作を繰り返し，⑥式が導入され補助問題の最適解Ⅰを得る．さらに2点問の線分上において有効条件式が存在しないためにその時点での点Ⅰが原問題の最適解となり表6が最適時タブローである．

2．4　目的関数値の収束傾向

シュミレーションによる条件式数200個，変数の数200個の線形計画問題（200×200）をRPPアルゴリズムとシンプレックス法によって解きその実験結果を図2に示す．縦軸に目的関数値，横軸に処理時間(sec)をとっている．RPPアルゴリズムは，初期補助問題として有効条件式数77個，初期目的関数値を∞とした問題から出発する．最初はシンプレックス法で解き補助問題の実行可能基底解を得る．この解が最適解でなければ，さらに有効条件式を選択するために目的関数の最大係数を指標として係数の大きな値から順にP個選び出す．ここでは，P＝1とするが問題の種類によりPの値は変わる．この操作を操り返しながら11回の選択により全部が最適解を拘束しないが，22個の条件式が導入され最適解に達している．曲線1は，RPPアルゴリズムの処理時時間に対する目的聞数値の収束性を示し，同様に曲線2は，シンプレックス法の収束性を示す．曲線1は，∞から最適解に収束し，そのLP問題における解の上限をあらわす．曲線2は，0から収束してゆき解の下限をあらわす．このことから曲線1から曲線2に下した垂線の長さは，目的関数値の差であり，最適解はこの垂線上に存在する．この差がなくなった時点での値が最適解を示す．RPPアルゴリズムの曲線1は時間の経過とともに収束してゆき，5.93(sec)，446 STEP 後，67個の条件式数で最適解Z＝586110.9570に達した．それに対しシンプレックス法の曲線2は，20.76(sec)，360 STEP 後に最適解に達している．この図からわかるようにRPPアルゴリズムは，シンプレックス法に比べ曲線1のような収束を示し，28.6%程度の処理時間で最適解に達していることがわかる．STEP回数を比べるとRPPアルゴリズムが446 STEP に対しシンプレックス法が360 STEP と

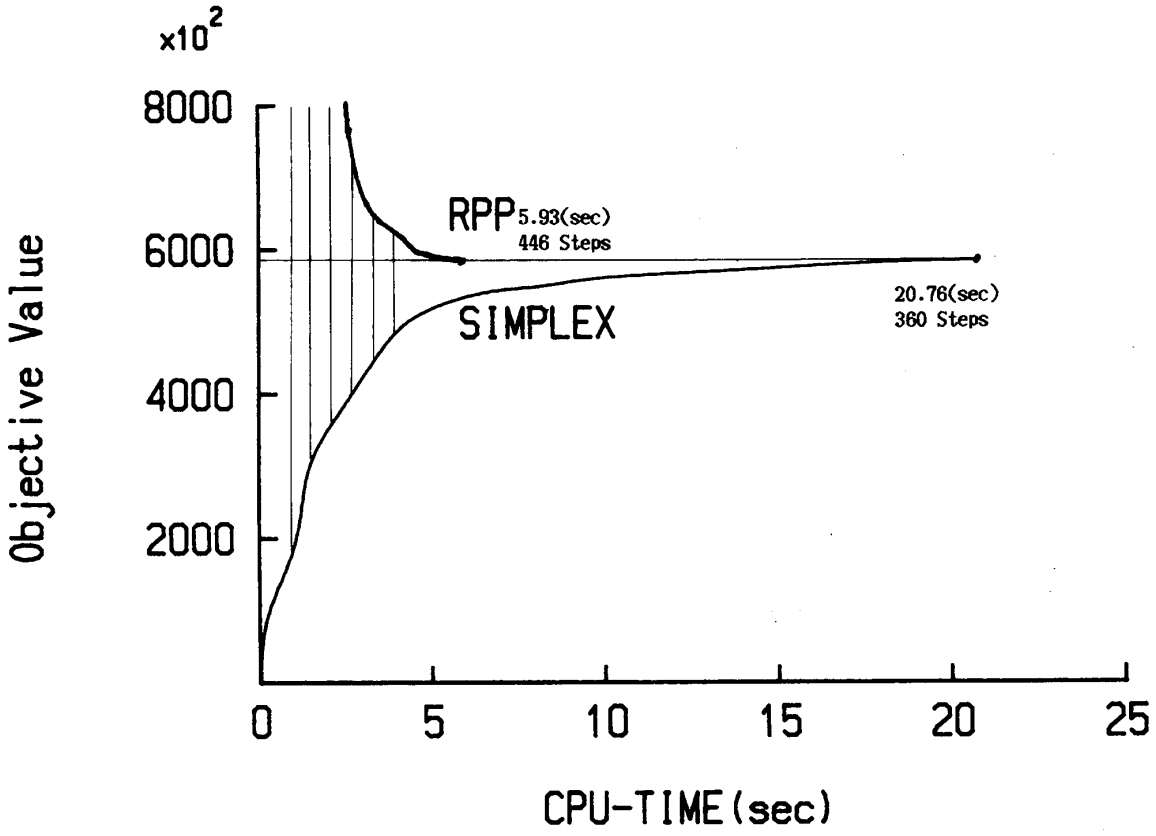

図2　単体法とRPPアルゴリズムの処理時間に対する目的数値の変化

RPP アルゴリズムの STEP 回数が多いが，初期条件式数45個から最終的に67個の条件式数により求解されるので，シンプレックス法のように全条件式についての掃き出し操作と比べると掃き出し要素の数が軽減されるためである．このシュミレーション問題での最適解に関与している条件式数200個の条件式数で求解される RPP アルゴリズムの有効性は明らかに示されている．

3．実験結果の比較検討

3，1　シンプレックス法との比較検討

本研究の計算実験で用いた LP 問題は，一様乱数によるシュミレーション問題である．条件式の各々の係数要素－10から10までをとり，定数は2500から8500まで，目的関数の係数要素は－30から30までをとる．係数要素に含まれる零は，全要素に対し20%，負の要素は残りの非零要素に対し10%とする．計算実験に使用したコンピュータは，富士通 M380である．

RPP アルゴリズムの有効性を調べるために，シュミレーションによる各種規模の問題を解きシンプレックス法と比較検討する．その実験結果は表7に示すとうりである．項目 a は，問題の規模をあらわし各欄での左の値が条件式数，右の値が変数の数を示している．本実験では70×50から1400×1300までの問題を考えている．項目 b，c はそれぞれ

a	PROBLEM SIZE	70×50	120×100	170×150	220×200	310×300	510×500	810×800	1100×1000	1400×1300
b	RPP CPU-TIME(sec)	0.96	1.32	2.36	3.76	8.54	37.04	135.85	320.51	780.36
c	SIMPLEX CPU-TIME(sec)	0.76	2.07	7.65	13.95	66.41	553.47	3631.57	9752.32	32928.59
d	CPU-TIME RATIO[RPP/SIM]	1.258	0.637	0.309	0.267	0.129	0.0669	0.0374	0.0329	0.0237
e	初期補助問題条件式数	15	19	25	29	36	45	57	67	76
f	最適時補助問題条件式数	21	24	34	39	48	70	87	100	120
g	f項 ／ 全条件式数	0.300	0.200	0.200	0.177	0.155	0.137	0.107	0.0909	0.0860
h	最適解を拘束する条件式数	12	14	21	24	31	47	59	76	86
i	h項 ／ 全条件式数	0.171	0.117	0.124	0.109	0.100	0.0922	0.0728	0.0691	0.0614

表7　各種問題の処理時間

RPPアルゴリズム，シンプレックス法の処理時間(sec)である．この表7からわかるように70×50規模の問題では，シンプレックス法が有利である事を示している．これは，RPPアルゴリズムが初期補助問題の作成や新たな有効条件式の選択とその導入等のオーバーヘッド時間が効いて，シンプレックス法が有利となっている．ところが，120×100規模になるとRPPアルゴリズムの有効性があらわれてくる．1400×1300規模の大規模問題において，項目b, cの処理時間を比較してみるとRPPアルゴリズムは780.36(sec)であるのに対しシンプレックス法は32928.59 (sec) と40倍の処理時間を要する．これを項目eの処理時間比でみてみると0.0237でありRPPアルゴリズムがシンプレックス法に対し2.37%程度の処理時間で最適解に達してかなり高い有効性を示した．この結果を

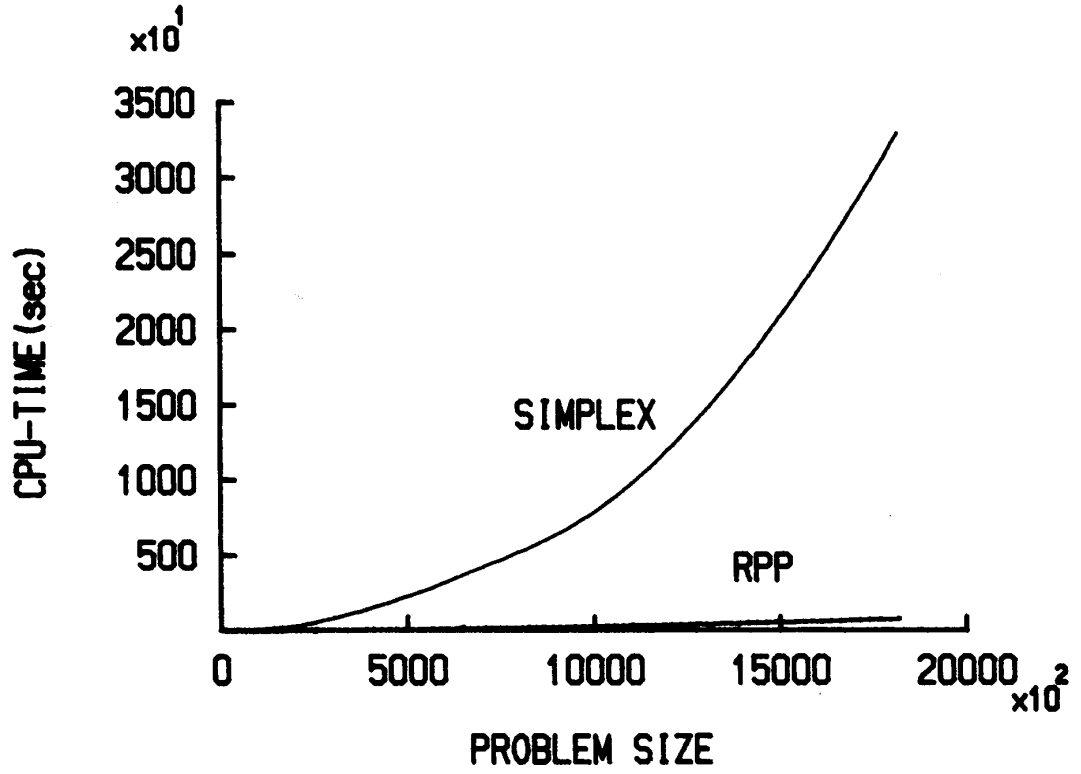

図3　各種問題に対する処理時間

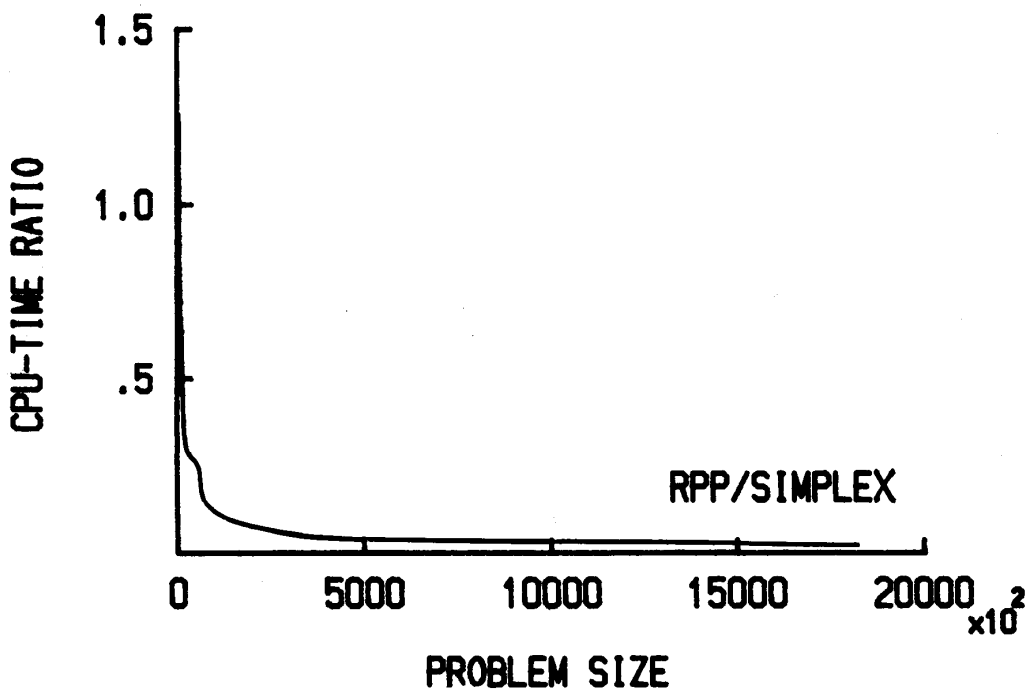

図4　各種問題に対するRPPアルゴリズムの有効性

グラフにしたものが図3，図4である．縦軸は，図3は処理時間(sec)，図4は処理時間比をとり，横軸に問題の規模をとる．問題の規模は要素の個数で示しており，例えば10×10の規模ならば100とする．このグラフから問題の大規模化に従い，シンプレックス法に対するRPPアルゴリズムの概要的な有効性がわかる．

このようなRPPアルゴリズムの効率化は，全条件式において最適解を拘束する条件式数が問題の大規模化にともないその割合が減少することから，補助問題の条件式数も減少してゆくことに起因していると思われる．この割合を項目gでみてみると70×50は全条件式数に対する割合が0.30であるが，大規模化に従いその割合が減少し，1400×1300規模においては0.0909となった．このことから，全条件式の各要素に対し掃き出しを行なうシンプレックス法に対し，補助問題としてRPPアルゴリズムは解いてゆくので掃き出し要素数が軽減され問題が大規模化するほどRPPアルゴリズムの効率が良くなり項目eはあきらかにその傾向を示している．

3．2　処理時間を最小にするPの選択

RPPアルゴリズムは，初期補助問題をシンプレックス法で解くことから出発して，その補助問題の実行可能基底解が最適解に達していなければ，有効条件式を選択し補助問題に付加してゆく，その選択には掃き出しにおいて目的関数の係数値が大きいほど目的関数値Zがより大きく増加することに注目し目的関数の係数を大きい順に選び出しそれに対応する軸からP個の実行可能領域を形成する有効条件式を抽出し補助問題に付

加する．その有効条件式を幾つ付加するか，つまりPの値をどれくらいに定めるかによってRPPアルゴリズムの処理時間が違っている．

このPの値に対する処理時間の実験結果を図5，6，7に示す．問題の規模は300×

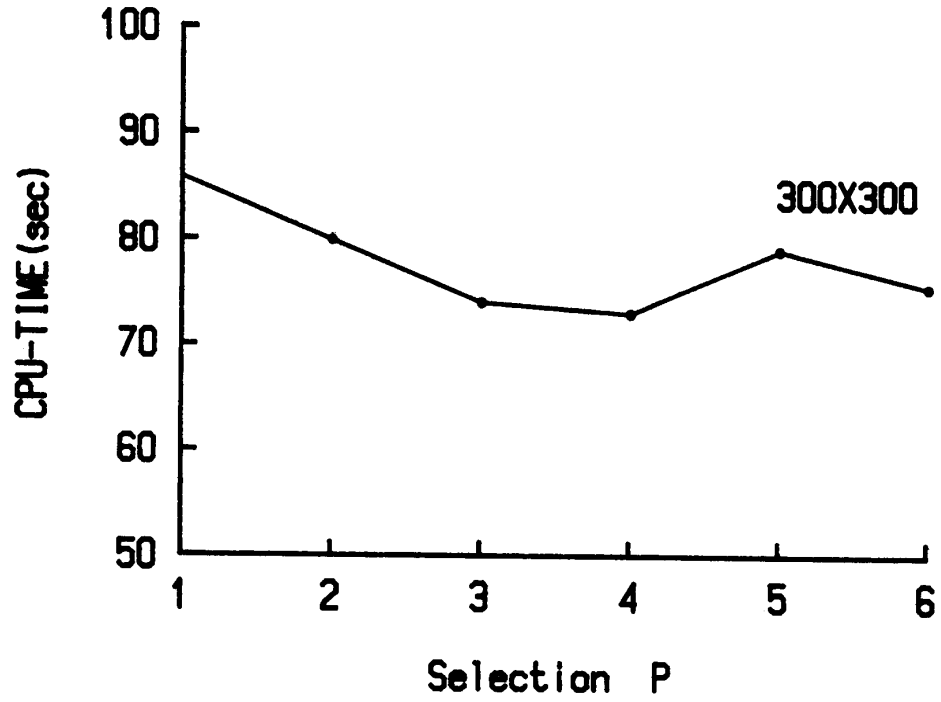

図5　300×300規模におけるPの値に対する処理時間

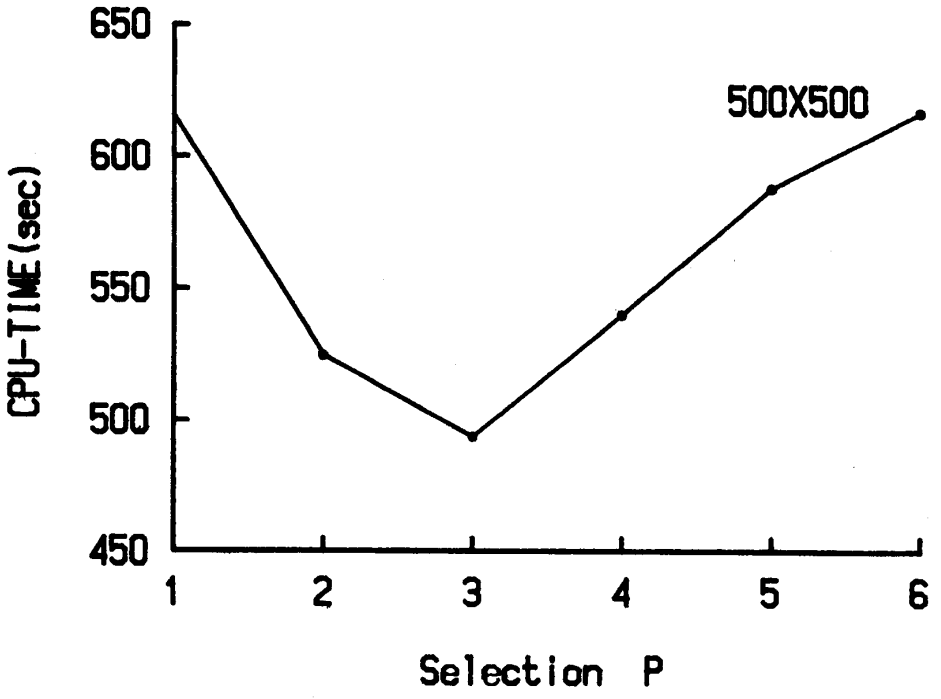

図6　500×500規模におけるPの値に対する処理時間

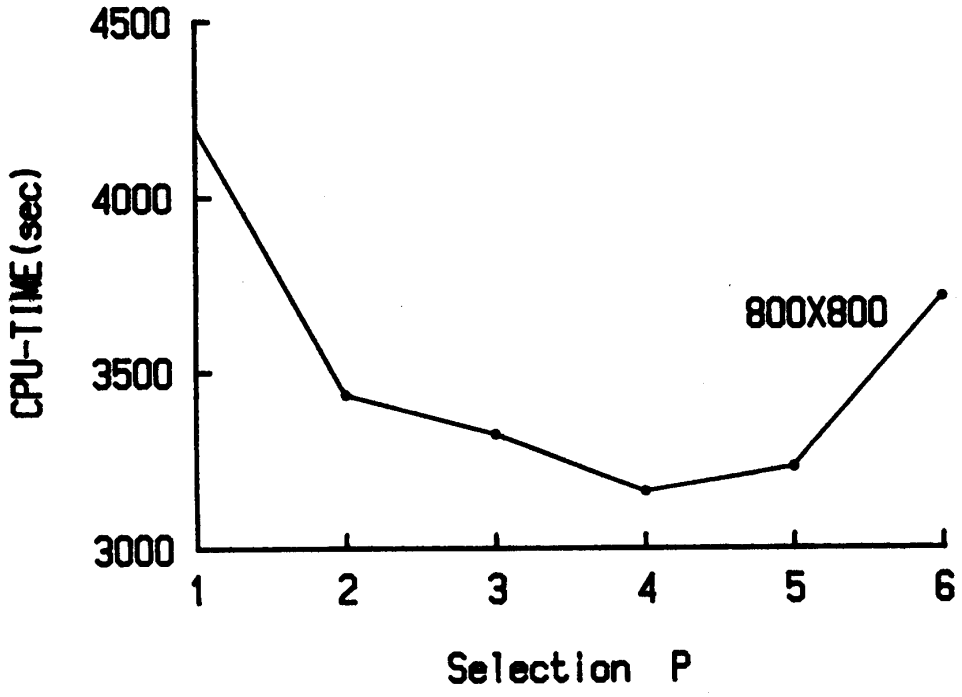

図7　800×800規模におけるPの値に対する処理時間

300, 500×500, 800×800の3種類の規模で零要素を60%, 負要素を40%, Pの値は0～5まで考えている. 図2からわかるようにPが0から増えるに従い処理時間が減少してゆくそしてある値Pから今度は増加傾向を示す. これは複数個の実行可能領域を形成する有効条件式が付加されることにより, 最適解を拘束する条件式が多く選択され, 補助問題への導入回数を減らしSTEP回数の軽減となるためである. しかし, Pを大きな値にとるとP個の有効条件式を選択するオーバーヘッド時間が効いてくる. 又実行可能領域を形成するが最適解を拘束しない有効条件式が増えそれに対する計算処理も効いてくるために処理時間が増加する.

以上のことから各種の問題に対する処理時間を最小とする最適なPはグラフに示すとうりP＝1～3が有効であると予想される.

4. 考察と今後の課題

本論文において, LP問題の新解法として有効条件式の抽出といった操作を行い原問題に対する部分問題においてシンプレックス法を用いて最適解を求解し, 処理時間の短縮をはかるRPPアルゴリズムを提案した. そしてシュミレーション問題における計算実験の結果は問題の大規模化に従い有効性が高くなり1400×1300規模の大規模問題においてはLPの代表的な解法であるシンプレックス法と問題の処理時間を比較して2.37%程度で最適解が求められかなり高い有効性を示すことが判明した. 又有効条件式を選択

する際各種規模に対してSTEP回数を減らすため複数個の条件式を導入するつまりP個の有効条件式を導入したときの処理時間を有効性を示した.

今後は，さらに最適解を拘束する条件式をいかに抽出するかについて検討する.

シュミレーション問題だけでなく，実際に企業などで解かれている具体的な問題についてもその有効性が表われるか行ないまたKlee-Mintyの問題の様なシンプレックス法では非常に処理時間がかかってしまう問題についても有効性が表われるか検討する．またこのRPPアルゴリズムとは違った観点により最適解を求解しようとする解法にKarmarkar法がある．これは1984年N.Karmarkar〔4〕が「LP問題を解くのに要する総計回数は問題の大きさのある多項式で抑えられる」という理論をもとにLP問題をNLP問題の一種とみなして内部法を用いて解くものである．この解法は，総計回数が低くく抑えられしかも処理時間が短いと主張しているが，実際に問題を解かせた場合いろいろなオーバーヘッド時間が考えられどの程度の有効性が現れるか明確でない．今後このkarmarkar法との比較検討も行なってゆきたい.

参考文献

〔1〕 Dantzig, G.B.: "Applications of the Simplex Method to Transportation Problems," in K-9 (1951).

〔2〕 Dantzig, G. B. and P. Wolfe: "The Decomposition Principle for Linear Programs," Operations Research 8, 101-111 (1960).

〔3〕 A. P. Sethi and G. L. Thompson: "The pivotand probe algorithm for solving a linear program," Mathematical Programming 29, 219-233 (1984).

〔4〕 Karmarkar, N.: "A New Polynomial-Time Algorithm for Linear Programming," ombinatorica 4,373-395 (1984).

〔5〕 A. P. Sethi and G. L. Thompson, "The non-candidate constraint method for reducing the size of a linear prgram", in: M. H. Karwan, V.Lofti, J. Telgen and S. Zionts, eds., Redundancy in Mathematical Programming (Springer, Berlin, Heidelberg, New York, Tokyo, 1983)

On the Efficiency of the RPP Algorithm for Solving a Large-Scale Linear Program

Kouyu UEMATSU* Toshio HYAKUTAKE**
and Hiroyuki NARIHISA**
**Information Processing Center*
***Department of Electronic Engineering*
Okayama University of Science
Ridaicho 1-1, Okayama 700. JAPAN
(Received September 30, 1987)

This paper presents an effective algorithm for solving a large-scale linear program. In a linear programming problem, the CPU time tends to increase remarkably according to the size of the problem. The RPP algorithm effectively candidates some constraints which constitute feasible solutions, because only about 5 to 20% of all constrants in the randomly generated problems will be tight at the optimum. We also present computational experience indicating that time savings of 84 ~ 97 % (problem size 220×220−1400×1300) over the simplex method can be obtained by the RPP algorithm.